The complexity of cyber attacks in a new layered-security model and the maximum-weight, rooted-subtree problem

Geir Agnarsson
Department of Mathematical Sciences
George Mason University
Fairfax, VA 22030
geir@math.gmu.edu

Raymond Greenlaw
Cyber Security Studies
United States Naval Academy
Annapolis, Maryland 21402
greenlaw@usna.edu

Sanpawat Kantabutra
Computer Engineering Department
Chiang Mai University
Chiang Mai, 50200, Thailand
sanpawat@alumni.tufts.edu

August 18, 2015

Abstract

This paper makes three contributions to cyber-security research. First, we define a model for cyber-security systems and the concept of a *cyber-security attack* within the model's framework. The model highlights the importance of *game-over components*—critical system components which if acquired will give an adversary the ability to defeat a system completely. The model is based on systems that use defense-in-depth/layered-security approaches, as many systems do. In the model we define the concept of *penetration cost*, which is the cost that must be paid in order to break into the next layer of security. Second, we define natural decision and optimization problems based on cyber-security attacks in terms of doubly weighted trees, and analyze their complexity. More precisely, given a tree T rooted at a vertex r, a *penetrating cost* edge function c on T, a *target-acquisition* vertex function p on T, the attacker's *budget* and the *game-over threshold* $B, G \in \mathbb{Q}^+$ respectively, we consider the problem of determining the existence of a rooted subtree T' of T within the attacker's budget (that is, the sum of the costs of the edges in T' is less than or equal to B) with total acquisition value more than the game-over threshold (that is, the sum of the target values of the nodes in T' is greater than or equal to G). We prove that the general version of this problem is intractable, but does admit a polynomial time approximation scheme. We also analyze the complexity of three restricted versions of the problems, where the penetration cost is the constant function, integer-valued, and rational-valued among a given fixed number of distinct values. Using recursion and dynamic-programming techniques, we show that for constant penetration costs an *optimal* cyber-attack strategy can be found in polynomial time, and for integer-valued and rational-valued penetration costs *optimal* cyber-attack strategies can be found in pseudo-polynomial time. Third, we provide a list of open problems relating to the architectural design of cyber-security systems and to the model.

Keywords: cyber security, defense-in-depth, game over, information security, layered security, weighted rooted trees, complexity, polynomial time, pseudo-polynomial time.

1 Introduction

Our daily life, economic vitality, and a nation's security depend on a stable, safe, and secure cyberspace. Cyber security is so important that the United States (US) Department of Defense established the US Cyber Command to take charge of pulling together existing cyberspace resources, creating synergy, and synchronizing war-fighting efforts to defend the information-security environment of the US [24]. Other countries also have seen the importance of cyber security. To name just a few in what follows, in response to North Korea's creation of a cyber-warfare unit, South Korea created a cyber-warfare command in December 2009 [23]. During 2010, China introduced its first department dedicated to defensive cyber war and information security in response to the creation of the US Cyber Command [4]. The United Kingdom has also stood up a cyber force [5]. Other countries are quickly following suit.

Cyberspace has become a new frontier that comes with new opportunities, as well as new risks. According to a 2012 study of US companies, the occurrence of cyber attacks has more than doubled over a 3-year period while the adverse financial impact has increased by nearly 40 percent [8]. More specifically, US organizations experienced an average of 50, 72, and 102 successful attacks against them per week in 2010, 2011, and 2012, respectively. In [21] a wide range of cyber-crime statistics are reported, including locations of attacks, motivation behind attacks, and types of attacks. The number of cyber attacks is increasing rapidly, and for the month of June 2013, 4% of attacks were classified as cyber warfare, 8% as cyber espionage, 26% as hacktivism, and 62% as cyber crime (see [21]). Over the past couple of years these percentages have varied significantly from month-to-month. In order to respond to cyber attacks, organizations have spent increasing amounts of time, money, and energy at levels that are now becoming unsustainable. Despite the amounts of time, money, and energy pouring into cyber security, the field is still emerging and widely applicable solutions to the problems in the field have not yet been developed.

A secure system must defend against all possible cyber attacks, including zero-day attacks that have never been known to the defenders. But, due to limited resources, defenders generally develop defense systems for the attacks that they do know about. Their systems are secure to known attacks, but then become insecure as new kinds of attacks emerge, as they do frequently. To build a secure system, therefore, requires first principles of security. "In other words, we need a *science of cyber security* that puts the construction of secure systems onto a firm foundation by giving developers a body of laws for predicting the consequences of design and implementation choices" [19]. To this end Schneider called for more models and abstractions to study cyber security [19]. In his article Schneider suggested building a science of cyber security from existing areas of computer science. In particular, he mentioned formal methods, fault-tolerance, cryptography, information theory, game theory, and experimental computer science. All of these subfields of computer science are likely to be valuable sources of abstractions and laws.

Cyber security presents many new challenges. Dunlavy et al. discussed what they saw as some of the major mathematical problems in cyber security [9]. One of the main challenges is modeling large-scale networks using explanatory and predictive models. Naturally, graph models were proposed. Some common measures of a graph that such a model would seek to emulate are distribution over the entire graph of vertex in-degrees and out-degrees, graph diameter, community structure, and evolution of any of the mentioned measures over time [6]. Pfleeger discussed a number of useful cyber-security metrics [17]. She introduced an approach to cyber-security measurement that uses a multiple-metrics graph as an organizing structure by depicting the attributes that contribute to overall security, and uses a process query system to test hypotheses about each of the goals based on metrics and underlying models. Rue, Pfleeger, and Ortiz developed a model-evaluation framework that involves making explicit each model's assumptions, required inputs, and applicability conditions [18].

Complexity science, which draws on biological and other natural analogues, seems under utilized, but perhaps is one of the more-promising approaches to understanding problems in the cyber-security domain [3]. Armstrong, Mayo, and Siebenlist suggested that models of complex cyber systems and their emergent behavior are needed to solve the problems arising in cyber security [3]. Additionally, theories and algorithms that use complexity analysis to reduce an attacker's likelihood of success are also needed. Existing work in the fields of fault tolerance and high-reliability systems are applicable too. Shiva, Roy, and Dasgupta proposed a cyber-security model based on game theory [20]. They discovered that their model works well for a dynamically-changing scenario, which often occurs in cyber systems. Those authors considered the interaction between the attacks and the defense mechanisms as a game played between the attacker and the defender.

This paper is our response to the call for more cyber-security models in [19]. This work also draws attention to the importance of designing systems that do not have *game-over components*—components that are so important that once an adversary has taken them over, one's system is doomed. Since, as we will see, such systems can be theoretically hacked fairly efficiently. We model (many known) security systems mathematically and then discuss their vulnerabilities. Our model's focus is on systems having layered security; each security layer possesses valuable assets that are kept in *containers* at different levels. An attacker attempts to break into these layers to obtain assets, paying penetration costs along the way in order to break in, and wins if a given game-over threshold is surpassed before the attacker's budget runs out. A given layer of security might be, for example, a firewall or encryption. The associated cost of by-passing the firewall or encryption is the penetration cost that is used in the model. We formalize the notion of a cyber attack within the framework of the model. For a number of interesting cases we analyze the complexity of developing cyber-attack strategies.

The outline of this article is as follows. In Section 2 we define the model for cyber-security systems, present an equivalent weighted-tree view of the model, and define natural problems related to the model. A general decision problem (Game-Over Attack Strategy, Decision Problem GOAS-DP) based on the model is proved NP-complete in Section 3; its corresponding optimization problem (GOAS-OP) is NP-hard. In sections 4, 5, and 6 we provide a polynomial-time algorithm for solving GOAS-OP when penetration costs are constant, a pseudo-polynomial-time algorithm for solving GOAS-OP when penetration costs are integers, a polynomial-time approximation algorithm for solving GOAS-OP in general, and a polynomial-time algorithm for solving GOAS-OP when penetration costs are rational numbers from a prescribed finite collection of possible rational costs, respectively. As an easy corollary, we obtain a pseudo-polynomial-time algorithm for solving an optimization problem on general weighted non-rooted trees. Table 1 summarizes the computational results of the paper. Conclusions and open problems are discussed in Section 7.

2 Model for Cyber-Security Systems

2.1 Basic Setup

When defining our cyber-security game-over model, we need to strike a balance between simplicity and utility. If the model is too simple, it will not be useful to provide insight into real situations; if the model is too complex, it will be cumbersome to apply, and we may get bogged down in too many details to see the forest from the trees. In consultation with numerous cyber-security experts, computer scientists, and others, we have come up with a good compromise for our model between ease-of-use and the capability of providing useful insights.

Many systems contain layered security or what is commonly referred to as *defense-in-depth*,

Problem Name	**Time**	**Class**
GOAS-DP	–	NP-complete
GOAS-OP	–	NP-hard
GOAS-DP constant pc	$O(m^2 n)$	P
GOAS-OP constant pc	$O(m^2 n)$	P
GOAS-DP integer pc	$O(B^2 n)$	pseudo-pt
GOAS-OP integer pc	$O(B^2 n)$	pseudo-pt
GOAS-OP approx.	$O((1/\epsilon)^2 n^3)$	P
GOAS-DP rational pc	$O(m^{2d} n)$	P
GOAS-OP rational pc	$O(m^{2d} n)$	P

Table 1: Summary of results about the cyber-security model contained in the paper. Note that in the table "pc" stands for "penetration cost," and "pseudo-pt" stands for pseudo-polynomial time. The values of m, n, B, and d are as given in the respective theorems.

where valuable assets are hidden behind many different layers or secured in numerous ways. For example, a *host-based defense* might layer security by using tools such as signature-based vendor anti-virus software, host-based systems security, host-based intrusion-prevention systems, host-based firewalls, encryption, and restriction policies, whereas a *network-based defense* might provide defense-in-depth by using items such as web proxies, intrusion-prevention systems, firewalls, router-access control lists, encryption, and filters [14]. To break into such a system and steal a valuable asset requires several levels of security to be penetrated. Our model focuses on this layered aspect of security and is intended to capture the notion that there is a cost associated with penetrating each additional level of a system and that attackers have finite resources to utilize in a cyber attack. We also build the concept of critical game-over components.

2.2 Definition of the Cyber-Security Game-Over Model

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$, $\mathbb{Q}$ be the rational numbers, and $\mathbb{Q}^+$ be the positive rational numbers. With the intuition provided in the previous section in mind, we now present the formal definition of the model.

Definition 2.1. *A* cyber-security game-over model *M is a six-tuple ($\mathcal{T}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{L}$, B, G), where*

1. *The set $\mathcal{T} = \{t_1, t_2, \ldots, t_k\}$ is a collection of* targets*, where $k \in \mathbb{N}$. The value k is the* number of targets*. Corresponding to each target t_i, for $1 \leq i \leq k$, is an associated* target acquisition value *$v(t_i)$, where $v(t_i) \in \mathbb{Q}$. We also refer to the target acquisition value as the* acquisition value *for short, or as the* reward *or* prize*.*

2. *The set $\mathcal{C} = \{c_1, c_2, \ldots, c_l\}$ is a collection of* containers*, where $l \in \mathbb{N}$. The value l is the* number of containers*. Corresponding to each container c_i, for $1 \leq i \leq l$, is an associated* penetration cost *$p(c_i)$, where $p(c_i) \in \mathbb{Q}$.*

3. *The set $\mathcal{D} = \{C_1, C_2, \ldots, C_l\}$ is the set of* container nestings*. The tuple C_i, for $1 \leq i \leq l$, is called the* penetration list *for container c_i and is a list in left-to-right order of containers that must be penetrated before c_i can be penetrated. If a container c_i has an empty penetration list, and its cost $p(c_i)$* has been paid*, we say that the* container has been penetrated*. If a container c_i has a non-empty penetration list and each container in its list has been* penetrated *in left-to-right order, and its cost $p(c_i)$ has been paid, we say that the* container has been penetrated.

The number of items in the tuple C_i is referred to as the depth of penetration required for C_i*. If container c_j appears in c_i's tuple C_i, we say that container c_i* is dependent on container c_j*. If there are no two containers c_i and c_j such that container c_i is dependent on container c_j and container c_j is dependent on container c_i, then we say the* model is well-formed*.*

4. *The set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ is a list of container names. These containers specify the* level-1 locations *of the targets. For $1 \leq i \leq k$ if target t_i has level-1 location l_i, this means that there is no other container $\widehat{c}$ such that container $\widehat{c}$ is dependent on container l_i and container $\widehat{c}$ contains target t_i. Target t_i is said to be* located at level-1 in container l_i*. The target t_i is also said to be* located in container l_i *or any container on which container l_i is dependent. When a target's level-1 container has been penetrated, we say that the* target has been acquired*.*

5. *The value $B \in \mathbb{Q}$ is the* attacker's budget*. The value represents the amount of resources that an attacker can spend on a cyber attack.*

6. *The value $G \in \mathbb{Q}$ is the* game-over threshold *signifying when critical components have been acquired.*

The focus of this paper is on cyber-security game-over models that are well-formed, which are motivated by real-world scenarios. In the next section we introduce a graph-theoretic version of the model using weighted trees.

Remarks: (i) In part 3 of the definition we refer to the cost of a container c_i being paid. By this we simply mean that $p(c_i)$ has been deducted from the remaining budget, B', and we require that $B' - p(c_i) \geq 0$. (ii) In part 4 of the definition we maintain a general notion of containment for targets by specifying the inner-most container in which a target is located. Although containers can have partial overlap, we require that the inner-most container be unique. In the next definition we formalize the notion of a *cyber-security attack strategy.*

Definition 2.2. *A* cyber-security attack strategy *in a cyber-security game-over model M is a list of containers $c_1, c_2, \ldots, c_r$ from M. The* cost of an attack strategy *is $\sum_{i=1}^{r} p(c_i)$. A* valid attack strategy *is one in which the penetration order is not violated. A* game-over attack strategy in a cyber-security game-over model *M is a valid attack strategy $c_1, c_2, \ldots, c_r$ whose cost is less than or equal to B and whose total target acquisition value $\sum_{i=1}^{r} v(t_i) \geq G$. We call such a game-over attack strategy in a cyber-security game-over model a* (successful) cyber-security attack *or* cyber attack *for short.*

Note that this notion of a cyber attack is more general than some, and, for example, espionage would qualify as a cyber attack under this definition. The definition does not require that a service or network be destroyed or disrupted. Since many researchers will think of Definition 2.1 from a graph-theory point of view, in the next section we offer that perspective. As we will soon see, the graph-theoretic perspective allows us to work more easily with the model mathematically and to relate to other known results.

2.3 Game-Over Model in Terms of Weighted Trees

In this section we describe the (well-formed) game-over model in terms of weighted trees. The set $\mathcal{D}$ of nested containers in Definition 2.1 has a natural rooted-tree structure, where each container corresponds to a vertex that is not the root, and we have an edge from a parent u down to a child v if and only if the corresponding container $c(u)$ includes the container $c(v)$ in it. The weight of an edge from a parent to a child represents the cost of penetrating the corresponding container. The weight of a vertex represents the acquisition value/prize/reward obtained by penetrating/breaking into that container.

Sometimes we do not distinguish a target from its acquisition value/prize/reward nor a container from its penetration cost. We can assume that the number of containers and targets is the same. Since if we have a container housing another container (and nothing else), we can just look at this "double" container as a single container of penetration cost equal to the sum of the two nested ones. Also, if a container contains many prizes, we can just lump them all into a single prize, which is the sum of them all. The following is a graph-theoretic version of Definition 2.1.

Definition 2.3. *A* cyber-security (game-over) model (CSM) *M is given by an ordered five tuple $M = (T, c, p, B, G)$, where T is a tree rooted at r having $n \in \mathbb{N}$ non-root vertices, $c : E(T) \to \mathbb{Q}$ is a penetration-cost weight function, $p : V(T) \to \mathbb{Q}$ is the target-acquisition-value weight function, and $B, G \in \mathbb{Q}^+$ are the attacker's budget and the game-over threshold value, respectively.*

Remarks: (i) Note that $V(T) = \{r, u_1, \ldots, u_n\}$, where r is the designated root that indicates the start of an attack. (ii) In most situations we have the weights c and p being non-negative rational numbers, and $p(r) = 0$.

Recall that in a rooted tree T each non-root vertex $u \in V(T)$ has exactly one parent. We let $e(u) \in E(T)$ denote the unique edge connecting u to its parent. For the root r, we let $e(r)$ be the empty set and $c(e(r))$ be 0. For a tree T with $u \in V(T)$, we let $T(u)$ denote the (largest) subtree of T rooted at u. It is easy to see the correspondence between Definitions 2.1 and 2.3. Analogously to Definition 2.2, we next define a *cyber-security attack strategy* in the weighted-tree model.

Definition 2.4. *A* cyber-security attack strategy (CSAS) *in a CSM $M = (T, c, p, B, G)$ is given by a subtree T' of T that contains the root r of T.*

- *We define the* cost *of a CSAS T' to be $c(T') = \sum_{u \in V(T')} c(e(u))$.*
- *We define a* valid CSAS (VCSAS) *to be a CSAS T' with $c(T') \leq B$.*
- *We define the* prize *of a CSAS T' to be $p(T') = \sum_{u \in V(T')} p(u)$.*

A game-over attack strategy (GOAS) *in a CSM $M = (T, c, p, B, G)$ is a VCSAS T' with $p(T') \geq G$. We sometimes refer to such a GOAS simply as a* cyber-security attack *or* cyber attack *for short.*

Note that in Definition 2.4 we use c (resp. p) to denote the total cost (respectively, total prize) of a cyber-security attack strategy. We also use c (resp. p) as the penetration-cost weight function (respectively, target-acquisition-value weight function). The overloading of this notation should not cause any confusion. Throughout the remainder of the paper, we will use Definitions 2.3 and 2.4.

2.4 Cyber-Attack Problems in the Game-Over Model

We now state some natural questions based on the CSM.

Problem 2.5. Given: *A cyber-security model $M = (T, c, p, B, G)$.*

- Game-Over Attack Strategy, Decision Problem (GOAS-DP):
  Is there a game-over attack strategy in M?
- Game-Over Attack Strategy, Optimization Problem (GOAS-OP):
  What is the maximum prize of a valid game-over attack strategy in M?

Needless to say, some special cases are also of interest, in particular, in Problems 2.5 when c is (i) a constant rational function, (ii) an integer-valued function, or (iii) takes only finitely many given rational values. We explore the general GOAS and these other questions in the following sections.

2.5 Some Limitations of the Model

Our model is a theoretical model. It is designed to give us a deeper understanding of cyber attacks and cyber-attack strategies. Of course, a real adversary is not in possession of complete knowledge about a system and its penetration costs. Nevertheless, it is interesting to suppose that an adversary is in possession of all of this information, and then to see what an adversary is capable of achieving under these circumstances. Certainly an adversary with less information could do no better than our fully informed adversary.

We are considering systems as they are. That is, we are given some system, targets, and penetration costs. If the system is a real system, we are not concerned about how to improve the security of that system per se. We assume that the system is already in a hardened state. We then examine how difficult it would be to attack such a system. We do not examine the question of implementations of a system. Our model can be used on any existing system. Some real systems will have more than one possible path to attack a target. And, in the future it may be worth generalizing the model to structures other than trees. The first step is to look at trees and derive some insight from these cases.

We have purposely chosen a target acquisition function which is simple. That is, we merely add together the total costs of the targets acquired. Studying this simple acquisition function is the first step. It may be interesting to study more-complex acquisition functions in the future. For example, one can imagine two targets that in and of themselves are of no real value, but when the information contained in the two are combined they are of great value. In some cases our additive function can capture this type of target depending on the structure of the model.

We describe the notion of a game-over component. In the model this concept is an abstract one. A set of components whose total value exceeds a given threshold comprise a "game-over component." A game-over component is not necessarily a single target although one can think of a high-cost target, which is included as a target in a set of targets that push us over the game-over threshold, as being the game-over component.

For easy reference, the following table contains our most common abbreviations, their spelled out meaning, and where they are defined.

CSM	cyber-security (game-over) model	Def. 2.3
CSAS	cyber-security attack strategy	Def. 2.4
VCSAS	valid cyber-security attack strategy	Def. 2.4
GOAS	game-over attack strategy	Def. 2.4
GOAS-DP	game-over attack strategy, decision problem	Def. 2.5
GOAS-OP	game-over attack strategy, optimization problem	Def. 2.5

Table 2: Abbreviations we use throughout the paper, all defined in this section.

3 Complexity of Cyber-Attack Problems

In this section we show that the general game-over attack strategy problems are intractable, that is, highly unlikely to be amenable to polynomial-time solutions. Consider a cyber-security attack model M, where T is a star centered at r having n leaves $u_1, \ldots, u_n$. Since each cyber-security attack T' of M can be presented as a collection $E' \subseteq E(T)$ of edges of T, and hence also as a collection of vertices $V' \subseteq V(T)$ by $T' = T[\{r\} \cup V']$, and vice versa, each collection of vertices

$V' \subseteq V(T)$ can be presented as $V' = V(T')$ for some cyber-security attack T' of M, and the GOAS-DP is exactly the decision problem of the 0/1-KNAPSACK PROBLEM [10], and the GOAS-OP is the optimization problem of the KNAPSACK PROBLEM. Note that the 0/1-KNAPSACK PROBLEM is usually stated using natural numbers as weights, but clearly the case for weights consisting of rational numbers is no easier to solve yet still in NP. So, we have the following observation.

Observation 3.1. *The* GOAS-DP *is NP-complete; the* GOAS-OP *is an NP-hard optimization problem.*

REMARK: Observation 3.1 answers an open question in the last section of [15], where it is asked whether or not the LST-TREE PROBLEM can be solved in polynomial time (we presume) for general edge lengths. Observation 3.1 is similar to [7, Theorem 2], where also a star is considered to show that their SUBTREEE is as hard as KNAPSACK.

Notice that the NP-completeness of GOAS-DP is a double-edge sword. It suggests that even an attacker who has detailed knowledge of the defenses of a cyber-security system would find the problem of allocating his (attack) resources difficult. On the other hand, the NP-completeness also makes it difficult for the defender to assess the security of his system. However, we will see in Section 5, that if we allow a slight proportional increase of the attacker's budget B to an amount of $(1 + \epsilon)B$ for an $\epsilon \geq 0$, then GOAS-OP admits a polynomial time approximation scheme, so it can be solved in time polynomial in n and $1/\epsilon$.

Sections 4, 5, and 6 consider the complexity of cyber-security attacks where c is a constant-valued cost function, an integer-valued cost function, and a rational-valued cost function of finitely many possible values, respectively. In Section 5, as mentioned, we also obtain an approximation algorithm for solving GOAS-OP, and a solution on general weighted non-rooted trees. In all cases we are able to give reasonably efficient algorithms for solving GOAS-OP.

4 Cyber Attacks with Constant Penetration Costs

In this section we show that if all penetration costs have the same value then the GAME-OVER ATTACK STRATEGY PROBLEMS can be solved efficiently in polynomial time. Consider a CSM M, where c is a constant function taking a constant rational value $c(e) = c$ for each $e \in E(T)$. That is, all penetration costs are a fixed-rational value. This variant is the first interesting case of the GOAS-DP and GOAS-OP, as there are related problems and solutions in the literature. One of the first papers on maximum-weight subtrees of a given tree with a specific root is [1], where it is shown that the *rooted subtree problem*, that is, to find a maximum-weight subtree with a specific root from a given set of subtrees, is in polynomial time if, and only if, the *subtree packing problem*, that is, to find maximum-weight packing of vertex-disjoint subtrees from a given set of subtrees (where the value of each subtree can depend on the root), is in polynomial time. In more-recent papers the *weight-constrained maximum-density subtree problem (WMSP)* is considered: given a tree T having n vertices, and two functions $l, w : E(T) \to \mathbb{Q}$ representing the "length" and "weight" of the edges, respectively, determine the subtree T' of T such that $\sum_{e \in E(T')} w(e) / \sum_{e \in E(T')} l(e)$ is a maximum, subject to $\sum_{e \in E(T')} w(e)$ having a given upper bound. In [13] an $O(w_{\max} n)$-time algorithm is given to solve the related, and more restricted, *weight-constrained maximum-density path problem (WMPP)*, as well as an $O(w_{\max}^2 n)$-time algorithm to solve the WMSP. In [15] an $O(nU^2)$-time algorithm is given for the WMSP, where U is the maximum total length of the subtree, and in [22] an $O(nU \lg n)$-time algorithm for the WMSP is given, which is an improvement in the case when $U = \Omega(\lg n)$. The WMSP has a wide range of practical applications. In particular, the related WMPP has applications in computational biology [13], and the related *weight-constrained* *least-density path problem (WLPP)* also has applications in computational biology, as well as in computer, traffic, and logistic network designs [15].

The WMSP is similar to our problem, and some of the same approaches used in [13], [15], and [22] can be applied in our case, namely the techniques of recursion and dynamic programming. There are not existing results that apply directly to our problems. Note that there is a subtle difference between our GOAS-OP and the WMSP, as a maximum-weight subtree (that is, with the prize $p(T')$ a maximum) might have low density and vice versa; a subtree of high density might be "small" with low total weight (that is, prize).

In [7] a problem on trees related to the Traveling Salesman Problem with profits is studied, which is similar to what we do. Both here and in [7] the most general form of the problems considered, in our case GOAS-DP in Observation 3.1 and in their case (as mentioned above) SubtreeE in [7, Theorem 2], are observed to be as hard as Knapsack and hence NP-complete. Also, the results of fixed costs, in our case Theorem 4.2 and in their case [7, Theorem 3], the problems are shown to be solvable in $O(n)$ time, given certain conditions. Theorem 4.2, however, provides a precise accounting for the time complexity and for certain values of m, defined there, our algorithm would be faster than that given in [7]. Their work is not in the context of cyber-security, and does not handle cases as general as this work.

For a CSM M, where c is a constant function, we first note that T' is a VCSAS if and only if $m = |E(T')| \leq \lfloor B/c \rfloor$. Hence, in this case the GOAS-OP reduces to finding a CSAS T' with at most m edges having $p(T')$ at a maximum. Note that if $m \geq n$, then the GOAS-OP is trivial since $T' = T$ is the optimal subtree. Hence, we will assume the budget B is such that $m < n$.

In what follows, we will describe our dynamic programming setup to solve GOAS-OP in this case. The core of the idea is simple: we construct a 2×2 matrix for each vertex u in the tree T that stores the maximum prize of a subtree rooted at u on at most k edges and that contains only the rightmost $d(u) - i + 1$ branches from u, for each $k \in \{1, \ldots, m\}$ and $i \in \{1, \ldots, d(u)\}$.

More specifically, we proceed as follows. We may assume that our rooted tree T has its vertices ordered from left-to-right in some arbitrary but fixed order, that is, T is a *planted plane tree*. Since T has $n \geq 1$ non-root vertices and $n+1$ vertices total, we know by a classic counting exercise [2] that the number of planted plane trees on $n+1$ vertices is given by the Catalan numbers C_n by obtaining a defining recursion for C_n by decomposing each planted plane tree into two rooted subtrees. Using this decomposition, we introduce some notation. For a subtree τ of T rooted at $u \in V(T)$ denote by $\tau(v)$ the largest subtree of τ that is rooted at a vertex v (if $v \in T[V(\tau)]$). Denote by u_ℓ the leftmost child of u in τ (if it exists). Let $\tau_\ell = \tau(u_\ell)$ denote the subtree of τ generated by u_ℓ, that is, the largest subtree of T rooted at u_ℓ. Finally, let $\tau'' = \tau - V(\tau_\ell) = T[V(\tau) \setminus V(\tau_\ell)]$ denote the subtree of τ generated by the vertices not in τ_ℓ. In this way we obtain a decomposition/partition of the planted plane tree τ into two vertex-disjoint subtrees τ_ℓ and τ'' whose roots are connected by a single edge $e(u_\ell)$. In particular, for each vertex $u \in V(T)$, we have a partition of $T(u)$ into $T(u)_\ell = T(u_\ell)$ and $T(u)''$, which we will denote by $T''(u)$ (that is $T(u)'' = T''(u)$). Note that if u is a leaf, then $T(u) = T''(u) = \{u\}$ and $u_\ell = T(u_\ell) = \emptyset$. Also, if u has exactly one child, which therefore is its leftmost child u_ℓ, then $T(u)$ is the two-path between u and its only child u_ℓ, $T''(u) = \{u\}$, and $T(u_\ell) = \{u_\ell\}$. Assuming the degree of u is $d(u)$, we can recursively define the trees $T^1(u), \ldots, T^{d(u)}(u)$ by

$$\begin{aligned} T^1(u) &= T(u), \\ T^{i+1}(u) &= (T^i)''(u). \end{aligned}$$

For each vertex $u \in V(T)$, we create a $d(u) \times (m+1)$ rational matrix as follows:

$$\mathbf{M}(u) = \begin{bmatrix} M_0^1(u) & M_1^1(u) & \cdots & M_m^1(u) \\ M_0^2(u) & M_1^2(u) & \cdots & M_m^2(u) \\ & & \vdots & \\ M_0^{d(u)}(u) & M_1^{d(u)}(u) & \cdots & M_m^{d(u)}(u) \end{bmatrix},$$

where $M_k^i(u)$ is the maximum prize of a subtree of $T^i(u)$ rooted at u with at most k edges for each $i \in \{1, \ldots, d(u)\}$ and $k \in \{0, 1, \ldots, m\}$. In particular, $M_0^i(u) = p(u)$ for each vertex u and $i \in \{1, \ldots, d(u)\}$. For each leaf u of T, and each i and k, we set $M_k^i(u) = p(u)$, and for each internal vertex u we have a recursion given in the following way: for a vertex u and an *arbitrary* subtree τ rooted at u, we let $M_k(u; \tau)$ be the maximum prize of a subtree of τ rooted at u having k edges or 0 if vertex u does not exist. If a maximum-prize subtree of τ with k edges does not contain the edge from u to its leftmost child u_ℓ, then $M_k(u; \tau) = M_k(u; \tau'')$. Otherwise, such a maximum subtree contains $i-1$ edges from τ_ℓ and $k-i$ edges from τ''. The following lemma is easy to show.

Lemma 4.1. *The arbitrary subtree τ rooted at u is a maximum-prize subtree with at most k edges that contains the leftmost child u_ℓ of u if and only if the included subtree of τ_ℓ is a maximum-prize subtree with at most $i-1$ edges rooted at u_ℓ and the included subtree of τ'' is a maximum-prize subtree with at most $k-i$ edges rooted at u for some $i \in \{1, \ldots, k\}$.*

By Lemma 4.1 we therefore have the following recursion:

$$M_k(u; \tau) = \max\left(M_k(u; \tau''), \max_{1 \le i \le k} \left(M_{i-1}(u_\ell; \tau_\ell) + M_{k-i}(u; \tau'') \right) \right). \tag{1}$$

Since now $M_k^i(u) = M_k(u; T^i(u))$ for each i and k, we see that we can compute each $M_k^i(u)$ from the smaller M's as given in (1) using $O(k)$-arithmetic operations. Because $k \in \{0, 1, \ldots, m\}$, this fact means in $O(m)$-arithmetic operations. Since we assume each arithmetic operation takes one step, we have that each $M_k^i(u)$ can be computed in $O(m)$-time given the required inputs. Therefore, $\mathbf{M}(u)$ can be computed in $d(u)m \cdot O(m) = d(u)O(m^2)$-time. Performing these calculations for each of the n vertices of our given tree T, we obtain by the Handshaking Lemma a total time of

$$t(n) = \sum_{u \in V(T)} d(u)O(m^2) = O(m^2) \sum_{u \in V(T)} d(u) = O(m^2)2(n-1) = O(m^2 n).$$

We finally compute a maximum prize VCSAS T' in M by $p(T') = M_m^1(r)$ for the root r of T. We conclude by the following theorem.

Theorem 4.2. *If $M = (T, c, p, B, G)$ is a CSM, where T has n vertices, c is a constant function, and $m = \lfloor B/c \rfloor$ then the* GOAS-OP *can be solved in $O(m^2 n)$-time.*

Remarks: (i) Note that Theorem 4.2 is similar to [7, Theorem 3]. (ii) Also note that the overhead constant is "small": for each vertex u, each k, and each i by (1) each of $M_k^i(u) = M_k(u; T^i(u))$ uses exactly $2k$ arithmetic operations, namely k additions and k comparisons. Hence, the exact number of arithmetic operations can, by the Handshaking Lemma, be given by

$$N(n, m) = \sum_{u \in V(T)} \sum_{k=0}^{m} d(u)(2k) = \sum_{u \in V(T)} d(u) \sum_{k=0}^{m} 2k = 2|E(T)|m^2 = 2(n-1)m^2.$$

We obtain an overhead constant of two. Since we assumed the budget given is such that $m < n$, we see that the GOAS-OP can be solved in $O(n^3)$ time.

Corollary 4.3. *The* GOAS-DP *when restricted to constant-valued penetration costs can be solved in $O(n^3)$ time and is in P.*

5 Cyber Attacks with Integer Penetration Costs and an Approximation Scheme

In this section we show that if all penetration costs are non-negative integers then the Game-Over Attack Strategy Problems can be solved in pseudo-polynomial time. We will then use that to obtain a polynomial time approximation algorithm.

5.1 Integer valued cost

Consider now a CSM $M = (T, c, p, B, G)$, where c is a non-negative integer-valued function, that is, $c(e) \in \{0, 1, 2, \ldots\}$ for each $e \in E(T)$. Note that we can contract T by each edge e with $c(e) = 0$, thereby obtaining a tree for our CSM M, where $c : E(T) \to \mathbb{N}$ takes only positive-integer values. We derive a polynomial-time algorithm in terms of n and B to solve the GOAS-OP. We can assume B is an integer here as well since otherwise we could just replace B with $\lfloor B \rfloor$. To produce our new algorithm we will tweak the argument given in Section 4 for the case when the cost function c is a constant.

Using the same decomposition of a subtree τ of T into u_ℓ and τ'' for our dynamic programming scheme, for each vertex u we will assign, as before, a $d(u) \times (B + 1)$ integer matrix as follows:

$$\mathbf{N}(u) = \begin{bmatrix} N_0^1(u) & N_1^1(u) & \cdots & N_B^1(u) \\ N_0^2(u) & N_1^2(u) & \cdots & N_B^2(u) \\ & & \vdots & \\ N_0^{d(u)}(u) & N_1^{d(u)}(u) & \cdots & N_B^{d(u)}(u) \end{bmatrix},$$

where $N_k^i(u)$ is the maximum prize of a subtree of $T^i(u)$ rooted at u of total cost at most k for each $i \in \{1, \ldots, d(u)\}$ and $k \in \{0, \ldots, B\}$. As before, we have $N_0^i(u) = p(u)$ for each vertex u. Similarly to Lemma 4.1, we obtain the following.

Lemma 5.1. *The arbitrary subtree τ rooted at u is a maximum-prize subtree of total cost at most k that contains the leftmost child u_ℓ of u if and only if the included subtree of τ_ℓ is a maximum-prize subtree of total cost at most $i - c(e(u_\ell))$ rooted at u_ℓ and the included subtree of τ'' is a maximum-prize subtree of total cost $k - i$ rooted at u, for some $i \in \{c(e(u_\ell)), \ldots, k\}$.*

Using similar notation and definitions as in Section 4, by Lemma 5.1 we get the following recursion:

$$N_k(u; \tau) = \max\left(N_k(u; \tau''), \max_{c(e(u_\ell)) \le i \le k} \left(N_{i-c(e(u_\ell))}(u_\ell; \tau_\ell) + N_{k-i}(u; \tau'')\right)\right), \tag{2}$$

and we obtain similarly the following.

Theorem 5.2. *If $M = (T, c, p, B, G)$ is a CSM, where T has n vertices and $c : E(T) \to \mathbb{N}$ takes only positive-integer values, then the* GOAS-OP *can be solved in $O(B^2 n)$-time.*

Remark: (i) Although we are not able to obtain a compact expression for the exact number of arithmetic operations that yield Theorem 5.2, the bound $N(n, B) = 2(n-1)B^2$ still is an upper bound, as for Theorem 4.2. (ii) Note the assumption that c is an *integer*-valued cost function is crucial, since otherwise, we would not have been able to use the recursion (2) in at most B steps.

Corollary 5.3. *The* GOAS-DP *when restricted to integer-valued penetration costs can be solved in pseudo-polynomial time.*

5.2 Approximation Scheme

We now can present a polynomial time approximation scheme (PTAS) for solving the GOAS-OP from Problem 2.5. In Observation 3.1 we saw that the GOAS-OP is an NP-hard optimization problem. But this is not the whole story; although it is hard to compute the exact solution, one can obtain a polynomial time approximation algorithm if we allow slightly more budget for the attacker than he/she wants to spend. We will in this section describe one such approximation scheme. Our approach here is similar to the PTAS for the optimization of the 0/1-Knapsack Problem presented in the classic text [16, Section 17.3].

We saw in Theorem 5.2 that GOAS-OP can be solved in $O(B^2 n)$-time, if the cost is integer valued and B is the budget of the attacker. So for large B this can be far polynomial time. For each fixed $t \in \mathbb{N}$ we can write the integer cost $c(e)$ of each edge $e \in E(T)$ as

$$c(e) = c_q(e) + c_r(e), \quad \text{where } c_r(e) = c(e) \bmod 2^t, \tag{3}$$

that is, we obtain a new cost function c_q by ignoring the last t digits of $c(e)$ when it is written as a binary number. Since each c_q is divisible by 2^t, solving GOAS-OP for c_q and budget B is the same as solving it for the cost function $2^{-t}c_q$ and budget $2^{-t}B$. Therefore, we can by Theorem 5.2 solve the GOAS-OP for this new cost function c_q in $O((2^{-t}B)^2 n)$-time.

Let T' (resp. T'_q) be an optimal GOAS-OP subtree of T w.r.t the cost c (resp. c_q), so $p(T')$ is maximum among subtrees with c-weight $\leq B$, and $p(T'_q)$ is maximum among subtrees with c_q-weight $\leq B$. In this case we have

$$c(T'_q) = c_q(T'_q) + c_r(T'_q) \leq B + |E(T'_q)| \cdot 2^t \leq B + n2^t. \tag{4}$$

Also, since $c_q(T') \leq c(T') \leq B$ we have by the definitions of T' and T'_q that $p(T') \leq p(T'_q)$. Therefore if there is a GOAS T' w.r.t. the cost c, then there certainly is one w.r.t. the cost c_q, namely T_q. Hence, if $\epsilon = \frac{n2^t}{B}$, then we obtain from (4) that $c(T_q) \leq (1+\epsilon)B$ and T'_q is here definitely a GOAS that further can be computed in $O((n/\epsilon)^2 n) = O((1/\epsilon)^2 n^3)$-time. Conversely, for a given $\epsilon \geq 0$, we obtain such an approximation algorithm by considering the cost c_q defined by (3) where

$$t = \left\lfloor \lg\left(\frac{\epsilon B}{n}\right)\right\rfloor. \tag{5}$$

We therefore have the following.

Theorem 5.4. *The* GOAS-OP *admits a polynomial time approximation scheme; for every* $\epsilon \geq 0$ *a GOAS* T' *of cost of at most* $(1+\epsilon)B$ *can be computed in* $O((1/\epsilon)^2 n^3)$*-time.*

Remarks: (i) In establishing the above Theorem 5.4 we started with an integer cost function $c : E(T) \to \mathbb{N}$. The same approach could have been used for a rational cost function $c : E(T) \to \mathbb{Q}$ where $c(e)$ has d binary binary digits after its binary point (i.e. radix point when written as a rational number in base 2.) By considering a new integer valued cost function $c' : E(T) \to \mathbb{N}$, where $c'(e) = 2^d c(e)$ for each $e \in E(T)$, we can in the same manner as used above, obtain an approximation algorithm where we replace B with $B' = 2^d B$. Needless to say however, in this case the corresponding cost function c'_q is obtained by truncating or ignoring only $t - d$ of the digits of c' (instead of the t digits of c), to obtain a solution using a budged of $(1+\epsilon)B$. (ii) Further along these lines, if the cost function $c : E(T) \to \mathbb{Q}$ is given as a fraction $c(e) = a(e)/b(e)$, where $a(e), b(e) \in \mathbb{N}$ are relatively prime, we can let M be the least common multiple of the $b(e)$ where $e \in E(T)$ and obtain by scaling by M a new integer valued cost function $c'' : E(T) \to \mathbb{N}$

where $c''(e) = Mc(e)$ for each $e \in E(T)$. Again, since c'' is integer valued we can in the same manner obtain an approximation algorithm where we replace B with $B'' = MB$. In this case the corresponding cost function c''_q is obtained by truncating or ignoring even fewer digits, namely $t - \lg M$ of the digits of c''. This will also yield a polynomial time approximation algorithm in terms of n and $1/\epsilon$ despite the fact that M can become very large (i.e. if all the costs have pairwise relatively prime denominators $b(e)$.)

5.3 General Weighted Trees

In our framework a CSM M is presented as a rooted tree provided with two weight functions: one on the vertices and one on the edges. In the model the root serves merely as a starting vertex and does not (usually) carry any weight (that is, has no prize attached to it). However, given a general non-rooted tree T provided with two edge-weight functions $w, w' : E(t) \to \mathbb{Q}$, we can always add a root to some vertex and then push the weights of one of the weight functions, say w down to the unique vertex away from the root. In this way we obtain a CSM M to which we can apply both Theorems 4.2 and 5.2. With this slight modification, we have the following corollary for general weighted trees.

Corollary 5.5. *Let T be a tree on n vertices, $w, w' : E(T) \to \mathbb{Q}$ two edge-weight functions, and B, G two rational numbers. If the function w is either (i) a rational constant $c \in \mathbb{Q}$ or (ii) integer-valued, then the existence of a subtree T' of T such that $w'(T') \leq B$ and $w(T')$ is a maximum can be determined in $O(m^2 n)$-time, where $m = \lfloor B/c \rfloor$ in case (i), and in $O(B^2 n)$-time in case (ii).*

6 Cyber Attack with Rational Penetration Costs

In this section we consider the more-general case of a CSM $M = (T, c, p, B, G)$ where the cost function $c : E(T) \to \mathbb{Q}$ takes at most d distinct rational values, say $c_1, \ldots, c_d \in \mathbb{Q}$. This case can model quite realistic scenarios, as there are currently only a finite number of known encryption methods and cyber-security designs, where a successful hack for each method/design has a specific penetration cost. As in previous sections, we will utilize dynamic programming and recursion based on the splitting of a subtree τ of a planted plane subtree into two subtrees τ_ℓ and τ'' as in (1) and (2). However, here we are dealing with rational-cost values (i.e. arbitrary *real* values from all practical purposes), and that the we are able to obtain a polynomial time procedure in this case is not as direct.

Note that if M is the least common multiple of all the denominators of $c_1, \ldots, c_d$, then by multiplying the cost and the budget of the attacker through by M, we obtain an integer valued cost function Mc, which then can by Theorem 5.2 be solved pseudo polynomially in $O(M^2 B^2 n)$-time. Our goal here in this section, however, is to develop an algorithm to solve GOAS-OP in time polynomial in n alone.

For each $i \in \{1, \ldots, d\}$, let $n_i = |\{e \in E(T) : c(e) = c_i\}|$, and so $\sum_{i=1}^{d} n_i = n = |E(T)| = |V(T)| - 1$. Let $\mathcal{B} = \{0, 1, \ldots, n_1\} \times \cdots \times \{0, 1, \ldots, n_d\} \subseteq \mathbb{Z}^d$, and note that $|\mathcal{B}| = \prod_{i=1}^{d}(n_i + 1)$. Denote a general d-tuple of $\mathbb{Q}^d$ by $\tilde{x} = (x_1, \ldots, x_d)$, and let $\tilde{x} \leq \tilde{y}$ denote the usual component-wise partial order $x_i \leq y_i$, for each $i \in \{1, \ldots, d\}$. If $\tilde{c} = (c_1, \ldots, c_d) \in \mathbb{Q}^d$ is the *rational-cost vector*, let $\mathcal{C} = \{\tilde{x} \in \mathbb{Q}^d : \tilde{x} \geq \tilde{0},\ \tilde{c} \cdot \tilde{x} \leq B\} \subseteq \mathbb{Q}^d$ denote the d-dimensional pyramid in $\mathbb{Q}^d$ with the $d + 1$ vertices given by the origin $\tilde{0} = (0, \ldots, 0)$ and $(0, \ldots, B/c_i, \ldots, 0)$, where $i \in \{1, \ldots, d\}$. To estimate the number of non-negative integral points in $\mathcal{C}$, we count the number of unit d-cubes within the pyramid $\mathcal{C}$. Since $\lfloor x \rfloor \leq x \leq \lfloor x \rfloor + 1$ for each rational x, then each $\tilde{x} \in \mathcal{C}$ is contained in the unit d-cube with the line segment from $\lfloor \tilde{x} \rfloor = (\lfloor x_1 \rfloor, \ldots, \lfloor x_d \rfloor)$ to $\lfloor \tilde{x} \rfloor + \tilde{1} = (\lfloor x_1 \rfloor + 1, \ldots, \lfloor x_d \rfloor + 1)$ as its diagonal. Since $\tilde{c} \cdot \tilde{x} \leq B$, then $\tilde{c} \cdot (\lfloor \tilde{x} \rfloor + \tilde{1}) \leq B + \sum_{i=1}^{d} c_i$, and hence, the number of integral points in $\mathcal{C}$ is at most the volume $\mathcal{V}(\mathcal{C}')$ of the associated pyramid $\mathcal{C}' = \{\tilde{x} \in \mathbb{Q}^d : \tilde{x} \geq \tilde{0},\ \tilde{c} \cdot \tilde{x} \leq B'\} \subseteq \mathbb{Q}^d$, where $B' = B + \sum_{i=1}^{d} c_i$, that is, at most $\lfloor \mathcal{V}(\mathcal{C}') \rfloor$, where

$$\mathcal{V}(\mathcal{C}') = \frac{1}{d!} \prod_{i=1}^{d} \frac{B'}{c_i} = \frac{1}{d!} \prod_{i=1}^{d} \left(\frac{B + \sum_{j=1}^{d} c_j}{c_i} \right).$$

Note that a CSAS T' of a CSM M has k_i edges of cost c_i for each i if and only if $\tilde{k} \in \mathcal{B} \cap \mathcal{C}'$.

Definition 6.1. *For each i let $m_i = \min(\lceil B'/c_i \rceil, n_i)$, and let $m = \sum_{i=1}^{d} m_i$.*

Remark: Note that we have $m = \sum_{i=1}^{d} m_i \leq \sum_{i=1}^{d} n_i = n$, and therefore any upper bound polynomial in m will yield a bound in the same polynomial in terms of n.

If $\mathcal{C}'' = \{0, 1, \ldots, \lceil B'/c_1 \rceil\} \times \cdots \times \{0, 1, \ldots, \lceil B'/c_d \rceil\}$, then $\mathcal{C}' \cap \mathbb{Z}^d \subseteq \mathcal{C}''$, and

$$\mathcal{B} \cap \mathcal{C}' = \mathcal{B} \cap (\mathcal{C}' \cap \mathbb{Z}^d) \subseteq \mathcal{B} \cap \mathcal{C}'' = \{0, 1, \ldots, m_1\} \times \cdots \times \{0, 1, \ldots, m_d\} \tag{6}$$

Hence, by the Inequality of Arithmetic and Geometric Mean (IAGM), we get

$$|\mathcal{B} \cap \mathcal{C}'| \leq |\mathcal{B} \cap \mathcal{C}''| = \prod_{i=1}^{d} (m_i + 1) \leq \left(\frac{\sum_{i=1}^{d} (m_i + 1)}{d} \right)^d = \left(\frac{m}{d} + 1 \right)^d.$$

We summarize in the following.

Observation 6.2. *If M is a CSM with n_i edges of cost c_i for each $i \in \{1, \ldots, d\}$, then $|\mathcal{B} \cap \mathcal{C}'| \leq (m/d + 1)^d$, which is a polynomial in $m = \sum_{i=1}^{d} m_i$ of degree d.*

Remark: Note that if $B'/c_i \leq n_i$ for each i, then $m_i = \min(\lceil B'/c_i \rceil, n_i) = \lceil B'/c_i \rceil$. In this case we have $\mathcal{C}' \cap \mathbb{Z}^d \subseteq \mathcal{B}$ and so $\mathcal{C}' \cap \mathbb{Z}^d = \mathcal{C}' \cap \mathbb{Z}^d \cap \mathcal{B} = \mathcal{C}' \cap \mathcal{B}$, and so again by the IAGM, we obtain

$$|\mathcal{B} \cap \mathcal{C}'| = |\mathcal{C}' \cap \mathbb{Z}^d| \leq \lfloor \mathcal{V}(\mathcal{C}') \rfloor = \left\lfloor \frac{1}{d!} \prod_{i=1}^{d} (m_i + 1) \right\rfloor \leq \left\lfloor \frac{1}{d!} \left(\frac{m}{d} + 1 \right)^d \right\rfloor,$$

where now $m = \sum_{i=1}^{d} \lceil B'/c_i \rceil$, which shows that, although polynomial in m of the same degree d as in Observation 6.2, the number of possible $\tilde{k} \in \mathcal{B} \cap \mathcal{C}'$ is a much smaller fraction of $(m/d + 1)^d$.

We now proceed with our setup for our dynamic programming scheme. As before, the idea is simple; we construct a multi-dimensional matrix/array for each vertex u of T, the construction of which is computed in a recursive manner, as for the previous 2×2 matrices $\mathbf{M}(u)$ and $\mathbf{N}(u)$.

Specifically, for each vertex u we assign a $d(u) \times |\mathcal{B} \cap \mathcal{C}'|$-fold array

$$\mathbf{A}(u) = \left[A^i_{\tilde{k}}(u) \right]_{\tilde{k} \in \mathcal{B} \cap \mathcal{C}',\ 1 \leq i \leq d(u)},$$

where $A^i_{\tilde{k}}(u)$ is the maximum prize of a subtree of $T^i(u)$ containing k_j edges of cost c_j for each $j \in \{1, \ldots, d\}$ and each $\tilde{k} \in \mathcal{B} \cap \mathcal{C}'$. For $\tilde{0} = (0, \ldots, 0)$, we have $A^i_{\tilde{0}}(u) = p(u)$ for each vertex u for $i = 1, \ldots, d(u)$.

Convention: For $i \in \{1, \ldots, d\}$ and an edge $e \in E(T)$, let $\delta_i(e) = \delta^{c(e)}_{c_i}$, where for every pair of rational numbers $x, y \in \mathbb{Q}$

$$\delta^y_x = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise} \end{cases}$$

is the *Kronecker delta function.* Further, let $\tilde{\delta}(e) = (\delta_1(e), \ldots, \delta_d(e))$.

As in (1) and (2), we use the same decomposition of a subtree τ of T into τ_ℓ and τ'', and as with previous Lemmas 4.1 and 5.1, we have the following.

Lemma 6.3. *The subtree τ rooted at u is a maximum-prize subtree among those with k_i edges of cost c_i for each i and that contains the leftmost child u_ℓ of u if and only if the included subtree of τ_ℓ is a maximum-prize subtree among those rooted at u_ℓ and with α_i edges of cost c_i for each i and the included subtree of τ'' is a maximum-prize subtree rooted at u among those that do not contain u_ℓ and with β_i edges of cost c_i for each i, for some $\tilde{\alpha}, \tilde{\beta} \in \mathcal{B} \cap \mathcal{C}'$, where $\tilde{\alpha} + \tilde{\beta} = \tilde{k} - \tilde{\delta}(e(u_\ell))$.*

For a vertex u and an arbitrary subtree τ rooted at u, we let $A_{\tilde{k}}(u;\tau)$ be the maximum prize of a subtree of τ rooted at u with k_i edges of cost c_i for each $i \in \{1,\dots,d\}$. If a maximum-prize subtree of τ with k_i edges of cost c_i does not contain the edge from u to its leftmost child u_ℓ, then $A_{\tilde{k}}(u;\tau) = A_{\tilde{k}}(u;\tau'')$. Otherwise, such a maximum subtree contains α_i edges of cost c_i from τ_ℓ and β_i edges of cost c_i from τ'', where $\alpha_i + \beta_i = c_i - \delta(e(u_\ell))$ for each $i \in \{1,\dots,d\}$. Finally, for each leaf u of T, each i, and $\tilde{k} \in \mathcal{B} \cap \mathcal{C}'$; we set $A^i_{\tilde{k}}(u) = p(u)$. As previously, we get by Lemma 6.3 the following recursion.

$$A_{\tilde{k}}(u;\tau) = \max\left(A_{\tilde{k}}(u;\tau''), \max_{\tilde{\alpha}+\tilde{\beta}=\tilde{k}-\tilde{\delta}(e(u_\ell))}\left(A_{\tilde{\alpha}}(u_\ell;\tau_\ell) + A_{\tilde{\beta}}(u;\tau'')\right)\right). \tag{7}$$

Lemma 6.4. *The evaluation of each $A^i_{\tilde{k}}(u)$ takes at most $2(m/d+1)^d$ arithmetic operations.*

Proof. For each $\tilde{x} = (x_1,\dots,x_d) \in \mathbb{Q}^d$, let $\pi^+(\tilde{x}) = \prod_{i=1}^d (x_i+1)$. By (7) each $A^i_{\tilde{k}}(u)$ requires $\pi^+(\tilde{k} - \tilde{\delta}(e(u_\ell)))$ additions and $\pi^+(\tilde{k} - \tilde{\delta}(e(u_\ell)))$ comparisons, and hence all in all $2\pi^+(\tilde{k} - \tilde{\delta}(e(u_\ell)))$ arithmetic operations.

By (6) we have that $\tilde{k} \in \mathcal{B} \cap \mathcal{C}' \subseteq \mathcal{B} \cap \mathcal{C}''$, and hence, $k_j \le m_j$ for each $j \in \{1,\dots,d\}$. Thus, by the IAGM, there are at most

$$2\pi^+(\tilde{k} - \tilde{\delta}(e(u_\ell))) < 2\prod_{j=1}^d (k_j+1) \le 2\prod_{j=1}^d (m_j+1) \le 2\left(\frac{m}{d}+1\right)^d$$

arithmetic operations for evaluating each $A^i_{\tilde{k}}(u)$. □

Assuming each arithmetic operation takes one step, the total running time to evaluate the entire array $\mathbf{A}(u)$ is at most a constant multiple of

$$\begin{aligned} N_d(n) &= \sum_{u\in V(T)} \sum_{\tilde{k}\in\mathcal{B}\cap\mathcal{C}'} \sum_{i=1}^{d(u)} 2\left(\frac{m}{d}+1\right)^d \\ &= \left(\sum_{u\in V(T)} d(u)\right)\left(\sum_{\tilde{k}\in\mathcal{B}\cap\mathcal{C}'} 2\left(\frac{m}{d}+1\right)^d\right) \\ &\le 2|E(T)|\left(\frac{m}{d}+1\right)^d 2\left(\frac{m}{d}+1\right)^d \\ &= 4(n-1)\left(\frac{m}{d}+1\right)^{2d}. \end{aligned}$$

We then obtain the desired maximum prize $p(T')$ of a VCSAS T' by $p(T') = \max_{\tilde{k}\in\mathcal{B}\cap\mathcal{C}'}\left(A^1_{\tilde{k}}(r)\right)$ for the root r of T of our CSM M, which takes at most $|\mathcal{B}\cap\mathcal{C}'| - 1 < (m/d+1)^d$ comparisons. Hence, we obtain the following.

Theorem 6.5. *If $M = (T, c, p, B, G)$ is a CSM where T has n vertices, m is given by Definition 6.1, and $c : E(T) \to \mathbb{Q}$ takes at most d distinct rational values, then the* GOAS-OP *can be solved in $O(m^{2d}n)$-time.*

Remarks: (i) Note that when $d = 1$, and hence $c_1 = c$, then m in Theorem 6.5 is given by $m = m_1 = \min(\lceil B'/c_1 \rceil, n) = \min(\lceil B/c \rceil + 1, n)$, whereas in Theorem 4.2 $m = \lceil B/c \rceil = \min(\lceil B/c \rceil, n)$, by the assumption that $\lceil B/c \rceil \leq n$. Still, the complexity when $d = 1$ in Theorem 6.5 clearly agrees with the complexity of $O(m^2 n)$ for solving the GOAS-OP when c is a constant function in Theorem 4.2. (ii) If each $m_i = O(f(n))$, for some "slow-growing" function of n, then Theorem 6.5 yields an $O(nf(n)^{2d})$-time algorithm for solving the GOAS-OP. In particular, if each $m_i = O(1)$, then Theorem 6.5 yields a linear-time in n algorithm to solve the GOAS-OP.

Corollary 6.6. *The* GOAS-DP *when restricted to d rational-valued penetration costs can be solved in polynomial time.*

7 Summary and Conclusions

This paper defined a new cyber-security model that models systems which are designed based on defense-in-depth. We showed that natural problems based on the model were intractable. We then proved that restricted versions of the problems had either polynomial time or pseudo-polynomial time algorithms. Table 1 in Section 1 summarizes our results. They suggest that in a real system the penetration costs should vary, that is, although each level should be difficult to attack, the cost of breaking into some levels should be even higher. The tree representation of the models suggests that systems should be designed to distribute targets in a bushy tree, rather than in a narrow tree. Most security systems are linear, and such systems could be strengthen by distributing targets more widely, providing *defense-in-deception.* Although in most situations a cyber attacker will not a priori know exact penetration costs, target locations, and prizes, the model still gives us insight into which types of security designs would be more effective.

We conclude the paper with a number of open questions.

1. Can we quantify how much targets need to be distributed in order to maximize security? For example, does an $(n + 1)$-ary tree provide provably better security than an n-ary tree?

2. Can we prove mathematically that the intuition of storing high-value targets deeper in the system and having higher penetration costs on the outer-most layers of the system results in the best security?

3. If targets are allowed to be repositioned periodically, what does that do to the complexity of the problems, and what is the best movement strategy for protecting targets?

4. Using the model, can one develop a set of benchmarks to rank the security of a particular system? How would one model prizes in a system?

5. Can the notion of time and intrusion detection be built into the model? That is, if an attacker tries to break into a certain container, the attacker may be locked out, resulting in game-over for that attacker, or perhaps may face an even higher new penetration cost.

6. Are there online variants of the model that are interesting to study? For example, a version where the topology of the graph changes dynamically or where only a partial description is known to the attacker.

Acknowledgments

This work was in part motivated by a talk that Bill Neugent of MITRE Corporation gave at the United States Naval Academy in the fall of 2011. We thank Bill for initial discussions about game-over issues relating to cyber-security models. Thanks also to Richard Chang for discussions about the model. – Finally, we like to thank the two anonymous referees for their careful reading of the paper, their pointed comments and suggestions which resulted in a greatly improved presentation of the results and made them more complete.

References

[1] El Houssaine Aghezzaf, Thomas L. Magnanti, and Laurence A. Wolsey. Optimizing Constrained Subtrees of Trees. *Mathematical Programming*, **71(2)**:113–126, Series A, (1995).

[2] Geir Agnarsson and Raymond Greenlaw. *Graph Theory: Modeling, Applications, and Algorithms*, Pearson Prentice Hall, Upper Saddle River, NJ, (2007).

[3] Robert C. Armstrong, Jackson R. Mayo, and Frank Siebenlist. Complexity Science Challenges in Cybersecurity, *Sandia Report*, March 2009.

[4] Tania Branigan. "Chinese Army to Target Cyber War Threat." *The Guardian (London).* `www.theguardian.com/world/2010/jul/22/chinese-army-cyber-war-department`, retrieved October 1, 2013.

[5] Hayes Brown. "No Longer in the Shadows, Cyberwar's Potential is now an Open Secret." *Think Progress.* `thinkprogress.org/security/2013/10/04/2699361/cyber-conflict-just-over-the-horizon/`, retrieved October 15, 2013.

[6] Deepayan Chakrabarti and Christos Faloutsos. Graph Mining: Laws, Generators, and Algorithms. *ACM Computing Surveys*, **38(1)**, article 2, 69 pages, (2006).

[7] Sofie Coene, Carlo Filippi, Frits Spieksma, and Elisa Stevanato. Balancing Profits and Costs on Trees. *Networks*, **61(3)**:200–11, (2013).

[8] "2012 Cost of Cyber Crime Study: United States," *Ponemon Institute*, research report, 29 pages, October 2012.

[9] Daniel M. Dunlavy, Bruce Hendrickson, and Tamara G. Kolda. Mathematical Challenges in Cybersecurity. *Sandia Report*, February 2009.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, (1979).

[11] Paul Goransson and Raymond Greenlaw. *Secure Roaming in 802.11 Networks*, Elsevier Science and Technical Book Group, (2007).

[12] Raymond Greenlaw, H. James Hoover, and Walter Larry Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*, *Oxford University Press*, (1995).

[13] Sun-Yuan Hsieh and Ting-Yu Chou. Finding a Weight-constrained Maximum-density Subtree in a Tree. *Algorithms and Computation, Lecture Notes in Computer Science*, **3827**:944–953, Springer, Berlin, (2005).

[14] Robert Johnston and Clint LaFever. Hacker.mil, Marine Corps Red Team (PowerPoint Presentation). (2012).

[15] Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. Finding a Length-constrained Maximum-sum or Maximum-density Subtree and Its Application to Logistics. *Discrete Optimization*, **3(4)**:385–391, (2006).

[16] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, Inc., (1982).

[17] Shari Lawrence Pfleeger. Useful Cybersecurity Metrics. *IT Professional*, **11(3)**:38–45, (2009).

[18] Rachel Rue, Shari Lawrence Pfleeger, and David Ortiz. A Framework for Classifying and Comparing Models of Cybersecurity Investment to Support Policy and Decision-making. *Proceedings of the Workshop on the Economics of Information Security*, 23 pages, (2007).

[19] Fred B. Schneider. Blueprint for a Science of Cybersecurity, *The Next Wave*, **19(2)**:47–57, (2012).

[20] Sajjan Shiva, Sankardas Roy, and Dipankar Dasgupta. Game Theory for Cyber Security. *Proceedings of the ACM* 6^{th} *Annual Cyber Security and Information Intelligence Research Workshop*, article no. 34, April 21–23, (2010).

[21] Paul Sparrows. Cyber Crime Statistics. `hackmageddon.com`, retrieved October 16, 2013.

[22] Hsin-Hao Su, Chin Lung Lu, and Chuan Yi Tang. An Improved Algorithm for Finding a Length-constrained Maximum-density Subtree in a Tree. *Information Processing Letters*, **109(2)**:161–164, (2008).

[23] Jung Sung-ki. "Cyber Warfare Command to Be Launched in January." *Koreatimes.co.kr.* `www.koreatimes.co.kr/www/news/nation/2013/07/205_56502.html`, retrieved October 1, 2013.

[24] William Jackson. "DOD Creates Cyber Command as U.S. Strategic Command Subunit." Federal Computer Week, `fcw.com`, October 16, 2013.